

# Veb aplikacija za elektronsku obradu dokumenata u MUP-u, odsek lična dokumenta

Miloš Simić

Fakultet tehničkih nauka

Univerzitet u Novom Sadu

simic.sf31.2019@uns.ac.rs

Sažetak: U ovom radu je opisan rad veb aplikacije za obradu dokumenata za izdavanje potvrda o važećim ličnim kartama i izdavanja i produživanja ličnih karti. Primenom predloženog rešenja se rešava problem prevelikih gužvi na šalterima prilikom zakazivanja termina i velikih dokumentacija prilikom izvršavanja ranije navedenih procesa. Za realizaciju su korišćene sledeće tehnologije: *SpringBoot* , *HTML5* , *CSS* , *MySQL Database*, *Java*, *JavaScript*, *React.js*.

Ključne reči: veb aplikacija, elektronska obrada dokumenata, lične karte, izdavanje, produživanje

## 1. Uvod

U današnjem digitalnom dobu, gde se sve više poslovnih i društvenih aktivnosti odvija putem interneta, efikasnost i brzina su ključne karakteristike koje nam omogućavaju da ostanemo konkurentni.

### Motivacija

Pristupanje administrativnim uslugama u vezi sa izdavanjem ličnih karti često zahteva dragoceno vreme građana. Dugotrajne procedure, nepotrebna čekanja i komplikovana dokumentacija postaju izvor frustracije za mnoge. Uvideli smo potrebu za modernizacijom ovog procesa kako bismo građanima omogućili brže i jednostavnije dobijanje ličnih karti, čime bi se smanjila birokratija i poboljšala efikasnost.

Problemi koje ovo rešenje rešava

Veb aplikacija za izdavanje ličnih karti ima za cilj da reši nekoliko ključnih problema. Prvo, omogućava građanima da podnesu zahtev za ličnu kartu iz udobnosti svog doma ili kancelarije, eliminisajući potrebu za fizičkim odlaskom na šaltere i čekanje u redovima. Ovo štedi vreme i olakšava svakodnevne obaveze građana. Drugo, aplikacija je dizajnirana sa naglaskom na jednostavnost i intuitivnost. Građani će biti vođeni kroz jasne korake i zahteve za dokumentaciju, čime se smanjuje mogućnost grešaka i nedoumica.

## 2. Srodna istraživanja

### 2.1. Srodna istraživanja

Istraživanja koja su obavljena prilikom izrade ovog rešenja su zasnovana na sajtu eUprave Republike Srbije. Sajt eUprave[7] Republike Srbije je ključna platforma koja omogućava građanima da obave različite administrativne procedure elektronskim putem. Na sajtu se može pronaći funkcionalnost zakazivanja termina za izdavanje lične karte, što olakšava građanima proces dobijanja ovog važnog identifikacionog dokumenta ali ne postoji funkcionalnost samog izdavanja lične karte ili dobijanja uverenja o važenju iste, koje implementiramo u ovom rešenju.

U zaključku, istraživanja o sajtu eUprave Republike Srbije za zakazivanje termina za izdavanje lične karte ukazuju na potrebu za unapređenjem i modernizacijom procesa. Implementacija funkcionalnosti elektronskog zakazivanja, mogućnosti prilaganja dokumenata u elektronskom formatu, samo izdavanje uverenja o važećoj ličnoj karti, izdavanje lične karte i mobilne kompatibilnosti bi poboljšala korisničko iskustvo građana, smanjila administrativne prepreke i efikasno unapredila proces izdavanja ličnih karata.

### 2.2. Korišćene tehnologije

U ovom odeljku navedene su tehnologije korišćene za razvoj rešenja predloženog u radu.

*SpringBoot*[1] - *SpringBoot* je okvir za razvoj web aplikacija koji se temelji na *Spring* okviru. On olakšava konfiguraciju i pokretanje aplikacija koristeći automatsku konfiguraciju, što znači da se mnoge osnovne konfiguracije aplikacije automatski izvršavaju bez potrebe za ručnim podešavanjem. *SpringBoot* takođe pruža podršku za različite vrste baza podataka, protokole za komunikaciju i druge tehnologije, što olakšava razvoj aplikacija. Pored toga, *SpringBoot* pruža i alate za lako testiranje i debugovanje aplikacije, što smanjuje vreme potrebno za razvoj i povećava efikasnost procesa. Ukratko, *SpringBoot* je okvir koji pomaže u razvoju web aplikacija, olakšavajući konfiguraciju i pokretanje aplikacije, te pružajući podršku za različite tehnologije i alate za testiranje i debugovanje.

*Java*[5] - *Java* je programski jezik koji je dizajniran da bude nezavisan od platforme i operativnog sistema. To znači da kod napisan u *Javi* može da se izvršava na različitim računarima, bez potrebe da se prilagođava svakom pojedinačnom sistemu. Ovaj jezik je razvijen od strane *Sun Microsystems* (sada je deo *Oracle-a*) 1994. godine i od tada je postao jedan od najpopularnijih jezika za razvoj aplikacija.

*Java* se često koristi za razvoj web aplikacija, mobilnih aplikacija, igara, aplikacija za poslovne potrebe i mnogih drugih vrsta softvera. To je objektno-orijentisan jezik i koristi se za koncepte kao što su nasleđivanje, polimorfizam i enkapsulacija.

Jedna od glavnih prednosti *Jave* je da ima veliku zajednicu programera koja razvija različite biblioteke i alate, što olakšava razvoj aplikacija. Takođe, *Java* ima robustan sistem za upravljanje memorijom, što omogućava da aplikacije rade stabilno i bez problema sa memorijom.

*MySQL*[4] - *MySQL* je popularni sistem za upravljanje bazama podataka koji se koristi širom sveta. Ovaj otvoreni izvor baze podataka relacionog tipa pruža efikasno skladištenje i manipulaciju podacima u aplikacijama. *MySQL* je laka za korišćenje i može se koristiti kao lokalna baza podataka ili kao baza podataka na mreži. Ovaj sistem podržava standardni *SQL* jezik i kompatibilan je sa različitim programskim jezicima kao što su *Java*, *C#* i *Python*.

*MySQL* baza podataka je napravljena da pruži visoke performanse i skalabilnost. Ona je dizajnirana za rad kao *in-memory* baza podataka, što znači da se podaci čuvaju u *RAM-u* umesto na disku. Ovo omogućava brzo čitanje i pisanje podataka, što je posebno korisno za aplikacije sa visokim nivoom pristupa podacima. Međutim, *MySQL* se takođe može konfigurisati da koristi disk za trajno skladištenje podataka, omogućavajući da se podaci sačuvaju čak i nakon prekida rada aplikacije.

Jedna od ključnih karakteristika *MySQL* baze podataka je mogućnost enkripcije podataka. Ovo obezbeđuje sigurno čuvanje podataka i zaštitu privatnosti korisnika. Takođe, *MySQL* ima konzolnu aplikaciju koja omogućava jednostavno upravljanje bazom podataka i izvršavanje *SQL* naredbi, olakšavajući administrativne zadatke i razvoj aplikacija.

*HTML5*[2] - *HTML (Hypertext Markup Language)* je jezik za označavanje koji se koristi za izradu web stranica. To je standardni jezik za izradu sadržaja na internetu i koristi se zajedno sa jezicima kao što su *CSS* i *JavaScript* za dizajniranje i funkcionalnost web stranica.

*HTML* se sastoji od oznaka (tagovi) koji se koriste za definisanje različitih elemenata na web stranici, kao što su naslovi, paragrafi, slike i linkovi. Ove oznake se koriste za definisanje strukture i sadržaja web stranice, a pregledači koriste te oznake da bi prikazali stranicu korisniku.

*HTML* takođe sadrži attribute, koji se koriste za definisanje dodatnih informacija o

oznakama. Na primer, atribut "*src*" se koristi za definisanje putanje do slike koja se treba prikazati na web stranici.

*HTML* se stalno razvija i nove verzije se redovno objavljuju. Trenutno se najčešće koristi *HTML5* verzija koja sadrži nove oznake i mogućnosti koje omogućavaju bolju interakciju i funkcionalnost web stranica.

Ukratko, *HTML* je jezik koji se koristi za izradu web stranica i sadržaja na internetu. On se koristi za definisanje strukture i sadržaja stranice, a pregledači ga koriste za prikazivanje stranice korisnicima.

*CSS*[3] - *CSS (Cascading Style Sheets)* je jezik za opisivanje izgleda i formatiranja web stranica. Koristi se zajedno sa jezikom *HTML* za dizajniranje web stranica i omogućava programerima da definišu kako će se pojedini elementi na stranici prikazivati.

*CSS* se sastoji od pravila koja se odnose na određene *HTML* oznake ili klase. Ova pravila definišu stilove kao što su boje, fontovi, margine i razmak između elemenata. Ova pravila se mogu definisati u samom *HTML* dokumentu ili se mogu učitati iz spoljnog fajla.

*CSS* takođe omogućava da se stilovi definišu za različite ekrane, što omogućava da se web stranice prikazuju na različitim uređajima kao što su desktop računari, tableti i mobilni telefoni.

*CSS* takođe poseduje mogućnosti za animaciju i transformaciju elemenata na stranici, što omogućava da se web stranice čine interaktivnijim i privlačnijim za korisnike.

Ukratko, *CSS* je jezik za opisivanje izgleda i formatiranja web stranica. On se koristi zajedno sa *HTML*-om za dizajniranje web stranica i omogućava programerima da definišu kako će se pojedini elementi na stranici prikazivati.

*JavaScript*[8] - *JavaScript* je programski jezik koji se izvodi na strani klijenta i koristi se za stvaranje interaktivnih veb stranica. Ovaj jezik je razvijen od strane *Netscape Communications Corporation* i prvi put je implementiran u *Netscape Navigator* pregledaču 1995. godine. Od tada je postao jedan od najvažnijih jezika za veb programiranje i podržan je na svim modernim veb pregledačima.

*JavaScript* se koristi za manipulaciju *HTML* elementima, upravljanje *CSS* stilovima, reagovanje na korisničke akcije i komunikaciju sa serverskom stranom putem *AJAX*-a. Ovaj jezik omogućava izgradnju dinamičnih veb stranica koje se ažuriraju bez potrebe za ponovnim učitavanjem cele stranice.

Jedna od glavnih prednosti *JavaScripta* je njegova jednostavnost i fleksibilnost. Ovaj jezik je objektno-orijentisan, ali podržava i funkcionalno programiranje. Takođe, *JavaScript* ima veliku zajednicu programera koja razvija različite biblioteke i okvire, kao što su *React*, *Angular* i *Vue.js*, koji olakšavaju razvoj složenih veb aplikacija.

*JavaScript* je takođe pogodan za razvoj mobilnih aplikacija putem okvira kao što je *React Native*. Ovaj okvir omogućava programerima da koriste *JavaScript* za izgradnju mobilnih aplikacija koje se izvode na *iOS* i *Android* platformama, koristeći zajednički kod.

*React*[9] - *React* je popularana *JavaScript* biblioteka za izgradnju korisničkih interfejsa. Ova biblioteka, koju je razvio *Facebook*, omogućava efikasno i reaktivno upravljanje prikazom veb stranica i aplikacija. *React* je nastao iz potrebe za efikasnijim upravljanjem velikim i kompleksnim korisničkim interfejsima. Tradicionalni pristup manipulaciji *DOM*-a (*Document Object Model*) putem direktne manipulacije elemenata je bio sklon performansnim i organizacionim problemima. *React* se fokusira na virtuelni *DOM* (*Virtual Document Object Model*) i reaktivni pristup, što omogućava brže i efikasnije ažuriranje prikaza. *React* koristi komponentni pristup u razvoju korisničkog interfejsa. Komponente su modularni delovi koda koji se mogu ponovno koristiti i koji se sastavljaju kako bi se kreirali složeni interfejsi. Ovo olakšava organizaciju koda, održavanje i testiranje.

Jedna od ključnih prednosti *Reacta* je njegova sposobnost upravljanja stanjem aplikacije. *React* koristi koncept "jedinstvenog izvora istine" (*Single Source of Truth*) i "prikazivanja" (*rendering*) da bi ažurirao samo one delove interfejsa koji zahtevaju promene. Ovo dovodi do efikasnijeg ažuriranja i poboljšane performanse.

*React* takođe podržava *JSX* sintaksu, koja omogućava pisanje *HTML* koda unutar *JavaScript* fajlova. Ovo olakšava razvoj korisničkog interfejsa, jer se *HTML* i *JavaScript* logika mogu kombinovati na intuitivan način.

*React* ima bogat ekosistem biblioteka i alata, kao što su *React Router* za upravljanje rutama, *Redux* za upravljanje stanjem, i mnogi drugi. Ove biblioteke i alati pružaju rešenja za različite aspekte razvoja veb aplikacija, čime olakšavaju i ubrzavaju proces razvoja.

### 3. Specifikacija zahteva

U ovom poglavlju su objašnjeni funkcionalni i nefunkcionalni zahtevi softverskog rešenja predstavljenog u ovom radu.

#### 3.1. Specifikacija funkcionalnih zahteva

U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za izdavanje ličnih karti i izdavanje potvrda o vazećim ličnim kartama.

Tabela 1 prikazuje opis slučaja korišćenja "Izdavanje uverenja o vazećoj ličnoj karti".

|            |   |
|------------|---|
| Naziv      | Izdavanje uverenja o vazećoj ličnoj karti   |
| Učesnici   | Službenik, Gradjanin  |
| Preduslovi | Slanje zahteva za izdavanje uverenja  |
| Koraci     | 1.Gradjanin salje zahtev za dobijanje uverenja, zahtev mora sadržati ime, prezime, JMBG i svrhu dobijanja uverenja<br>2.Službenik proverava zahtev i izdaje uverenje ako postoji važeća lična karta za dati JMBG iz zahteva |
| Rezlutat   | Gradjaninu je uspesno dostavljeno uverenje  |
| Izuzeci    | Zahtev ne ispunjava preduslove  |

Tabela 1- Opis slučaja korišćenja "Izdavanje uverenja o vazećoj ličnoj karti"

Tabela 2 prikazuje opis slučaja korišćenja “Kreiranje nove lične karte”.

|            |   |
|------------|---|
| Naziv      | Kreiranje nove lične karte                    |
| Učesnici   | Službenik                                     |
| Preduslovi | -   |
| Koraci     | 1.Službenik kreira novu ličnu kartu           |
| Rezultat   | Službenik je uspesno kreirao novu ličnu kartu |
| Izuzeci    | -   |

Tabela 2- Opis slučaja korišćenja “Kreiranje nove lične karte”

Tabela 3 prikazuje opis slučaja korišćenja “Pregled svih ličnih karti”.

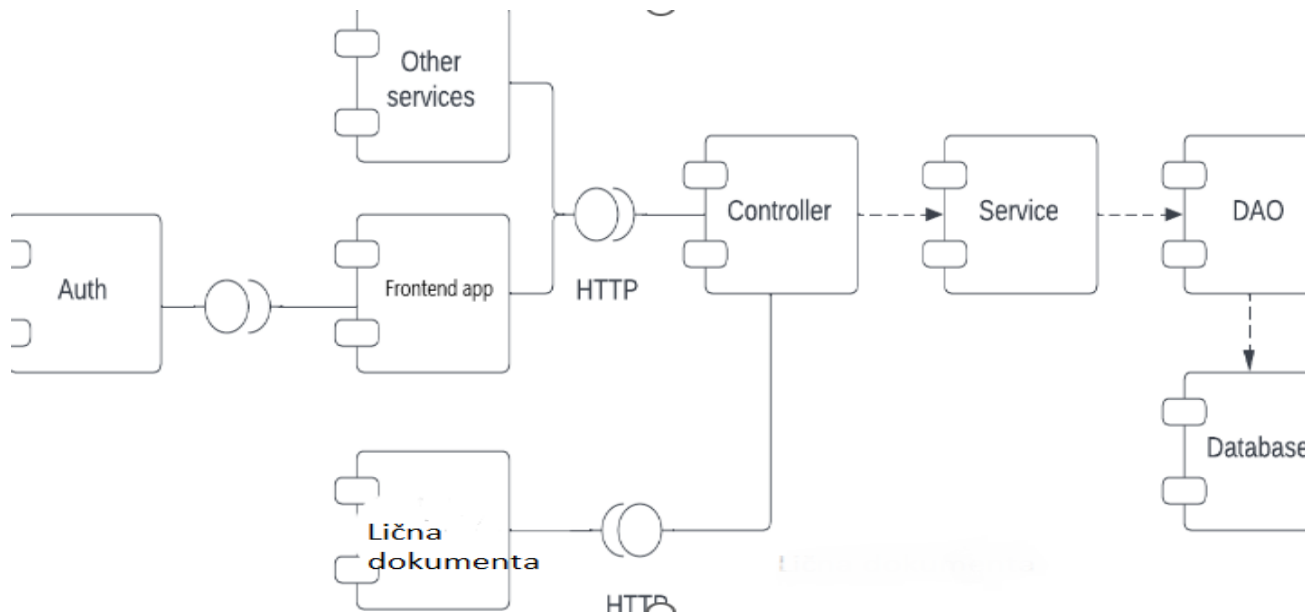
|            |   |
|------------|---|
| Naziv      | Pregled svih ličnih karti   |
| Učesnici   | Službenik   |
| Preduslovi | -   |
| Koraci     | 1.Službenik klikom na link prelazi na stranicu sa tabelarnim prikazom svih ličnih karti |
| Rezultat   | Službenik je uspesno pregledao lične karte  |
| Izuzeci    | -   |

Tabela 3- Opis slučaja korišćenja “Pregled svih ličnih karti”



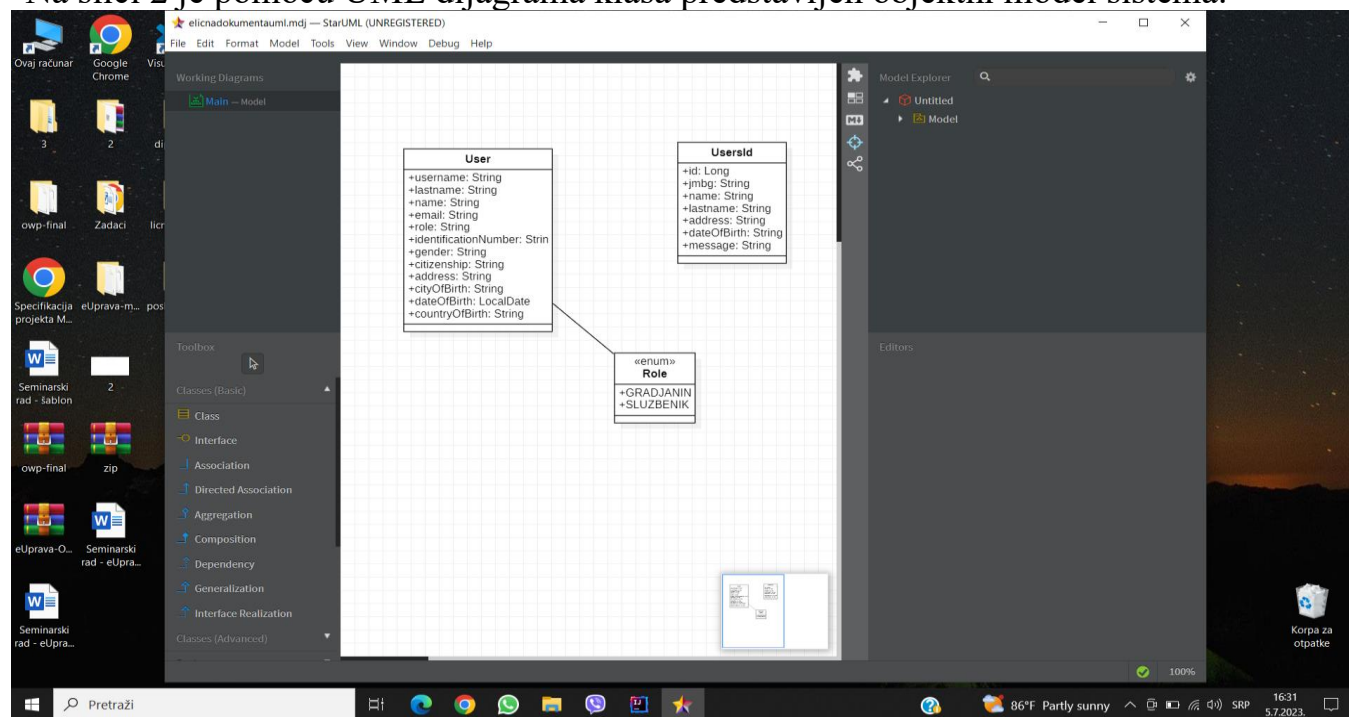
## 4. Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za kreiranje lične karte kao i izdavanja uverenja o važećoj ličnoj karti i pregleda svih ličnih karti u nastavku prikazano na slici 1.



Slika 1 - component diagram

Na slici 2 je pomoću UML dijagrama klasa predstavljen objektni model sistema.



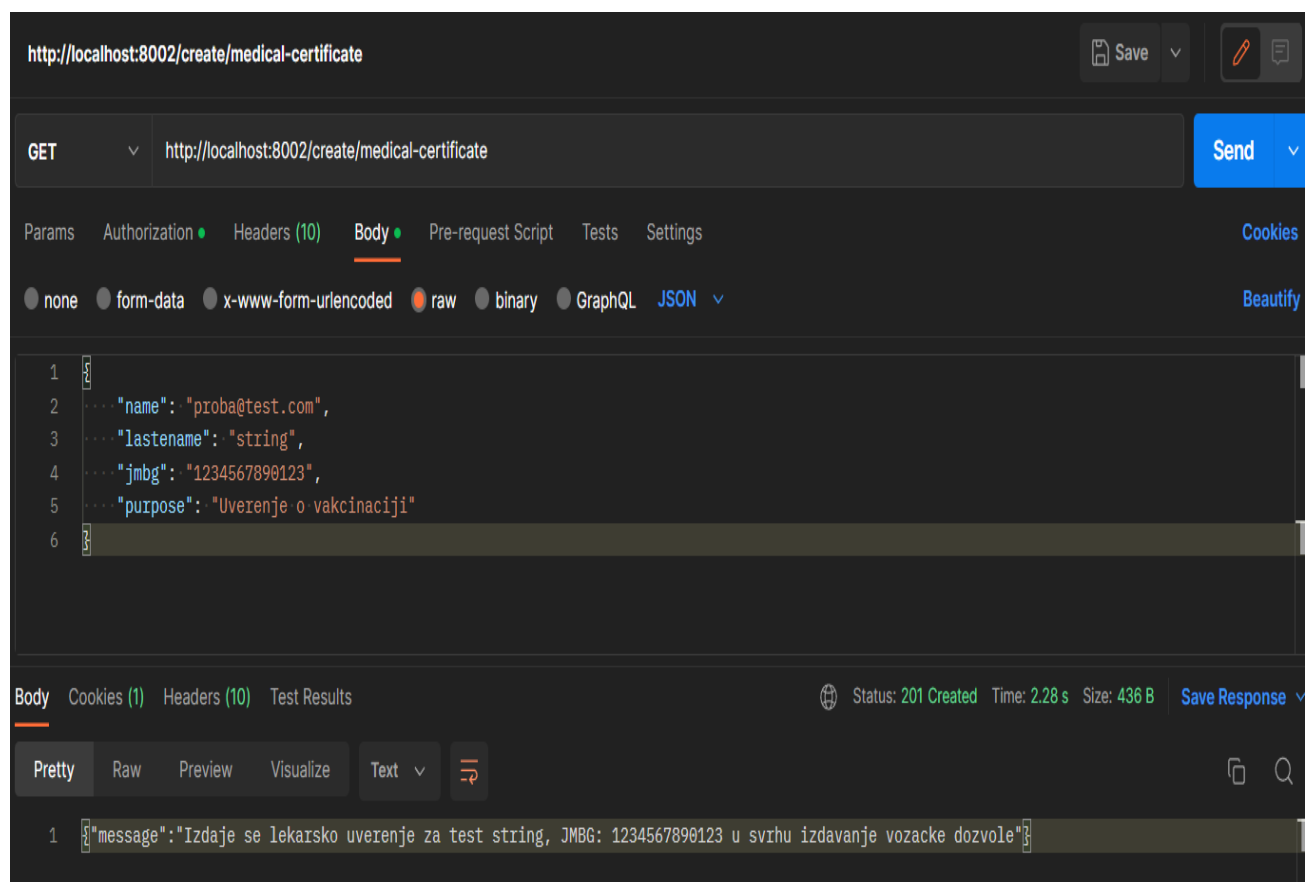
Slika 3 - class diagram





Klasa Gradjanin reprezentuje korisnike sistema i sadrži njihove lične podatke i podatke za autentifikaciju na sistem.

U nastavku ovog poglavlja, prikazano na slici broj 4, prikazana je komunikacija koju drugi sistemi obavljaju sa ovim, putem sistema koji je implementiran, drugi sistemi kada stupe u vezu sa ovim šalju popunjeni zahtev za dobijanje uverenja o važećoj ličnoj karti. U zahtevu navode ime, prezime, jmbg i svrhu za rad koje im je potrebno uverenje. Kao odgovor na validan zahtev dobijaju uverenje.



Slika 4

## 5. Implementacija

Ovo poglavlje prikazuje način na koji su implementirane neke od funkcija sistema za slanje zahteva za nabavku oružja.

Objašnjena je implementacija serverskog I klijentskog dela aplikacije.

### 5.1. Serverski deo aplikacije

Za prijavu korisnika na sistem implementira je metoda prikazana na listingu 1.

```
@RequestMapping(value = "/login", method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> login(@RequestBody LoginDTO user) {
    logger.info("Login attempt by user {}", user.getUsername());
    try {
        UsernamePasswordAuthenticationToken token = new UsernamePasswordAuthenticationToken(user.getUsername(),
            user.getIdentificationNumber());
        Authentication authentication = authenticationManager.authenticate(token);
        SecurityContextHolder.getContext().setAuthentication(authentication);
        UserDetails details = userDetailsService.loadUserByUsername(user.getUsername());
        User userDb = userService.findByUsername(user.getUsername());
        return new ResponseEntity<>(HttpStatus.OK);
    } catch (Exception e) {
        logger.debug("Nevalidni podaci");
        e.printStackTrace();
        return new ResponseEntity<>(new ErrorDTO("Nevalidni podaci"), HttpStatus.BAD_REQUEST);
    }
}
```

Listing 1 - Prijava korisnika na sistem

Ovoj metodi se prosledjuje predefinisani *body LoginDTO* koji reprezentuje konkretne podatke korisnika koji se proveravaju prilikom autentifikacije prijave. *AuthenticationManager* proverava dobijeni token, da bi onda servis proverio da li u bazi postoji korisnik sa unetim korisničkim imenom (na prikazanom listingu korisničko ime=*Username*), u slučaju postojanja korisnika sa tačnim korisničkim



imenom metoda vraća *HttpStatus.OK* I korisnik je prijavljen na sistem, u slučaju nevalidnih podataka metoda vraća *HttpStatus.BAD\_REQUEST* I korisniku ispisuje poruku “Nevalidni podaci”.

Za kreiranje nove lične karte implementirana je metoda na listingu 2.

```
@PostMapping("/createIDforUser")
public ResponseEntity<UserId> save(@RequestBody UserId userId) {
    userIdService.save(userId);
    return new ResponseEntity<>(userId, HttpStatus.CREATED);
}
```

Listing broj 2

Na ovom listingu prikazana je *post* metoda za kreiranje nove lične karte, metodi je prosledjen *@RequestBody* u vidu klase *UserId* iz modela. Nakon toga poziva se metoda *save* iz servisa koja prima podatke prosledjene od strane korisnika, obradjuje podatke, čuva ličnu kartu u bazi i vraća odgovor o uspešnom kreiranju.

Na sledecem listingu, listingu broj 3, prikaza je metoda za prikaz svih ličnih karti.

```
@GetMapping("/findAllIDs")
public ResponseEntity<UserId> findAllIDs(UserId userId) {
    userIdService.findAll(userId);
    return new ResponseEntity<>(userId, HttpStatus.OK);
}
```

Listing broj 3

Prikazana je *get* metoda za dobijanje liste svih ličnih karti. Kao parametar prosledjuje se klasa *UserId*, nakon čega se poziva metoda *findAll* iz servisa koja pristupa bazi podataka, pretražuje je i nakon pronadjenih podataka prosledjuje nam te podatke kao i poruku o uspešnom statusu.



Na narednom listingu, listingu broj 4, prikazan je kontroler sa metodom za kreiranje i izdavanje uverenja o važećoj ličnoj karti.

```
@CrossOrigin
@RestController
@Validated
@RequestMapping("/api/id-certificates")
public class IDCertificateController {

    @Autowired
    private UsersIdService usersIdService;

    @PostMapping
    public ResponseEntity<IDResponse> generateIDCertificateForUser(
        @RequestBody @Valid IDRequest request
    ){
        return ResponseEntity.ok(usersIdService.generateID(request));
    }

    @GetMapping("/{jmbg}")
    public ResponseEntity<CheckUserIDResponse> checkIfUserHasCertificate(@PathVariable String jmbg) {
        return ResponseEntity.ok(usersIdService.checkUserIDResponse(jmbg));
    }
}
```

Listing broj 4

Na prikazanom listingu imamo *post* metodu za generisanje uverenja, metodi se prosledjuje *@RequestBody IDRequest* koji u sebi sadrži potrebna polja za generisanje uverenja. Nakon što metoda dobije popunjen zahtev ona vraća status OK i poziva metodu iz servisa koja generise i šalje uverenje. Takođe možemo videti i *get* metodu kojom proveravamo da li je zahtev za uverenje koji je korisnik poslao validan.

## 6. Demonstracija

Ovo poglavlje prikazuje način korišćenja aplikacije za službenika.

Nakon što se službenik uspešno prijavio, aplikacija ga vodi na sledecu stranicu na kojoj se prikazuje stranica za izdavanje ličnih karti, prikazano na slici 5.

Izdavanje ličnih karti

Name: Milos

Last Name: Simic

JMBG: 12121212125

Address: Sabac, Jevremovac, 7. Okt

Date of Birth: 01.06.2023

Message: poruka

Submit

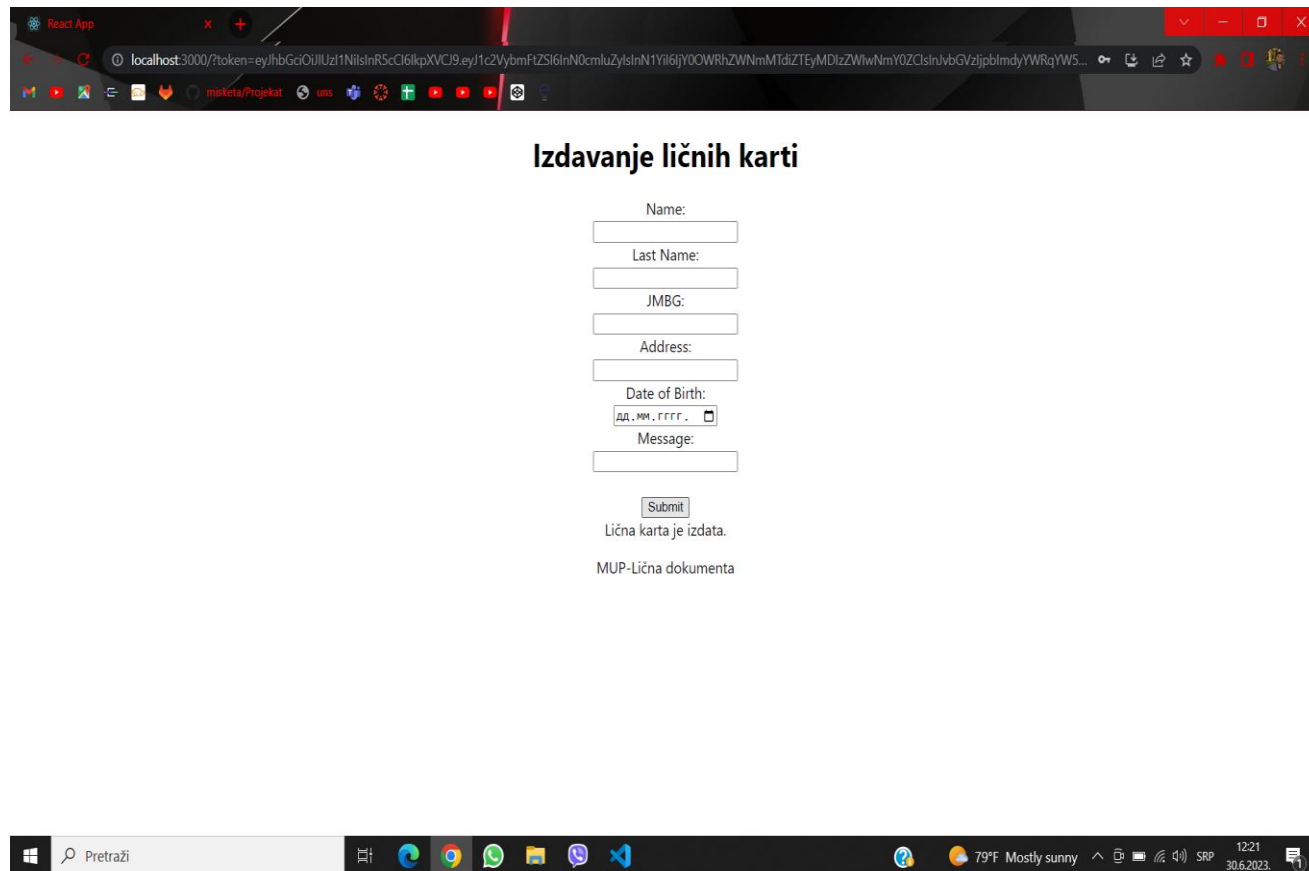
MUP-Lična dokumenta

Slika 5 - Prikaz stranice za izdavanje ličnih karti

Na stranici je prikazana forma koju službenik popunjava podacima korisnika kojem izdaje ličnu kartu. Nakon popune klikom na dugme “submit” kreira se lična karta ako su svi podaci validni.

Nakon toga korisnik dobija praznu formu i poruku koja potvrđuje uspešno izdavanje.

Na slici 6 prikazuje se stranica sa porukom o uspešnom izdavanju.



The screenshot shows a web browser window with the title "React App" and a URL starting with "localhost:3000". The page content is centered and features the heading "Izdavanje ličnih karti" in bold. Below the heading is a form with the following fields: "Name:", "Last Name:", "JMBG:", "Address:", "Date of Birth:" (with a date picker showing "dd.mm.yyyy."), and "Message:". A "Submit" button is located below the form. Below the button, the text "Lična karta je izdata." and "MUP-Lična dokumenta" are displayed. The browser's taskbar at the bottom shows the Windows logo, a search bar with "Pretraži", and several application icons. The system tray on the right indicates a temperature of 79°F, "Mostly sunny" weather, and the date/time "12:21 30.6.2023."

### Izdavanje ličnih karti

Name:

Last Name:

JMBG:

Address:

Date of Birth:

Message:

Lična karta je izdata.  
MUP-Lična dokumenta

Slika 6 - Prikaz stranice sa porukom

Na slici 7 prikazan je stranica sa tabelom u kojoj su izlistane sve lične karte u sistemu.

Slika 7 – Prikaz stranice sa tabelom svih ličnih karti







## 7. Zaključak

Zaključak ovog seminarskog rada o veb aplikaciji za izdavanje ličnih karti uverenja o važećim ličnim kartama ukazuje na mnoge prednosti ovog pristupa. Aplikacija se pokazala kao efikasan i praktičan način za pojednostavljenje procesa izdavanja ličnih dokumenata i smanjenje administrativnih troškova.

Kroz veb aplikaciju, korisnicima je omogućeno da podnesu zahtev za izdavanje lične karte ili uverenja o važećoj ličnoj karti putem online forme. Ovo ubrzava proces komunikacije između zahtevaoca i nadležnih organa, smanjujući vreme čekanja i mogućnost grešaka. Automatizacija nekih koraka procesa izdavanja takođe doprinosi efikasnosti i brzini realizacije zahteva.

Pored toga, aplikacija pruža transparentnost i pristupačnost celokupnog procesa izdavanja ličnih dokumenata. Ovo omogućava nadležnim organima da provere ispravnost zahteva i zakonitost izdavanja, čime se suzbija potencijalna korupcija. Osim toga, korisnici imaju uvid u status svog zahteva i mogu pratiti napredak njegove obrade.

Iako je aplikacija još u fazi testiranja, rezultati dosadašnjih ispitivanja su veoma obećavajući. Ova veb aplikacija predstavlja značajan korak napred u unapređenju procesa izdavanja ličnih karti i uverenja o važećim ličnim kartama. Uvođenje ovog sistema može doneti brojne benefite, uključujući smanjenje administrativnih troškova, bržu i efikasniju obradu zahteva i poboljšanje kvaliteta usluga.

Važno je napomenuti da ovaj sistem trenutno ne podržava elektronsko plaćanje prilikom izdavanja ličnih dokumenata, već se plaćanje vrši putem uplate na račun. Međutim, ovaj nedostatak se može prevazići implementacijom odgovarajuće platforme za elektronska plaćanja.

U poređenju sa sličnim rešenjima, ovaj sistem se ističe jednostavnošću korišćenja i pružanjem svih neophodnih funkcionalnosti. Iako postoji mogućnost da druga analizirana rešenja imaju određene prednosti koje proizlaze iz angažovanja timova inženjera, verujemo da je ovaj sistem i dalje konkurentan i pruža odgovarajuću podršku korisnicima. Trenutno, pristup ovom sistemu je moguć putem veb pregledača naračunarima sa Windows operativnim sistemom. U budućnosti, daljim razvojem sistema, trebalo bi omogućiti podršku za korišćenje sistema i na Android i iOS operativnim sistemima. Ovo bi proširilo dostupnost aplikacije i omogućilo korisnicima da pristupe uslugama izdavanja ličnih dokumenata putem mobilnih uređaja.

U zaključku, veb aplikacija za izdavanje ličnih karti i uverenja o važećim ličnim kartama predstavlja inovativno i efikasno rešenje koje olakšava proces izdavanja ličnih dokumenata, smanjuje troškove i povećava transparentnost. Njeno dalje unapređivanje i proširenje funkcionalnosti omogućiće još veću efikasnost i prilagodljivost korisnicima. Ovaj sistem predstavlja važan korak napred ka modernizaciji administrativnih procesa i pružanju kvalitetnih usluga građanima.





## 8. Literatura

- [1] - <https://spring.io/projects/spring-boot>
- [2] - <https://www.hostinger.com/tutorials/what-is-html>
- [3] - <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [4] - [MySQL](#)
- [5] - [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html)
- [6] - [Sajt Ministarstva unutrašnjih poslova](#)
- [7] - [eUprava](#)
- [8] - [JavaScript](#)
- [9] - [React.js](#)